quanteda

September 16, 2014

R topics documented:

`bigrams` *Create bigrams*

Description

Create bigrams

Usage

```
bigrams(text, window = 1, concatenator = "_", include.unigrams = FALSE,
  ...)
```

Arguments

`text`	character vector containing the texts from which bigrams will be constructed
`window`	how many words to be counted for adjacency. Default is 1 for only immediately neighbouring words. This is only available for bigrams, not for ngram.
`concatenator`	character for combining words, default is _ (underscore) character
`include.unigrams`	if `TRUE`, return unigrams as well
`...`	additional arguments passed to tokenize

Value

a character vector of bigrams

Author(s)

Kohei Watanabe and Ken Benoit

Examples

```
bigrams("The quick brown fox jumped over the lazy dog.")
bigrams("The quick brown fox jumped over the lazy dog.", window=2)
```

`collocations` *Detect collocations in a text*

Description

returns a list of collocations. Note: Currently works only for pairs (bigram collocations).

Usage

```
collocations(text = NULL, file = NULL, top = NA, distance = 2, n = 2,
  method = c("lr", "chi2", "mi"))
```

Arguments

`text`	a text or vector of texts
`file`	a filename containing a text
`top`	threshold number for number of collocations to be returned (in descending order of association value)
`distance`	distance between pairs of collocations
`method`	association measure for detecting collocations
`n`	Only bigrams (n=2) implemented so far.

Value

A list of collocations, their frequencies, and their test statistics

Author(s)

Kenneth Benoit

Examples

```
data(inaugCorpus)
collocations(texts(inaugCorpus)[1], top=50)
collocations(texts(inaugCorpus)[1], top=50, method="chi2")
```

corpus *Constructor for corpus objects*

Description

Creates a corpus from a document source, such as character vector (of texts), or an object pointing to a source of texts such as a directory containing text files. Corpus-level meta-data can be specified at creation, containing (for example) citation information and notes.

Usage

```
corpus(x, ...)

## S3 method for class directory
corpus(x, enc = NULL, docnames = NULL,
  docvarsfrom = c("filenames", "headers"), docvarnames = NULL, sep = "_",
  source = NULL, notes = NULL, citation = NULL)

## S3 method for class character
corpus(x, enc = NULL, docnames = NULL, docvars = NULL,
  source = NULL, notes = NULL, citation = NULL, ...)

is.corpus(x)
```

Arguments

`x`	A source of texts to form the documents in the corpus. This can be a filepath to a directory containing text documents (see directory), or a character vector of texts.
`docvarsfrom`	Argument to specify where docvars are to be taken, from parsing the filenames (filenames) separated by `sep` or from meta-data embedded in the text file header (`headers`).
`docvarnames`	Character vector of variable names for `docvars`
`sep`	Separator if docvar names are taken from the filenames.
`docnames`	Names to be assigned to the texts, defaults to the names of the character vector (if any), otherwise assigns "text1", "text2", etc.
`docvars`	A data frame of attributes that is associated with each text.
`source`	A string specifying the source of the texts, used for referencing.
`notes`	A string containing notes about who created the text, warnings, To Dos, etc.

Details

`is.corpus` returns `TRUE` if the object is a corpus

Value

A corpus class object containing the original texts, document-level variables, document-level metadata, corpus-level metadata, and default settings for subsequent processing of the corpus. A corpus consists of a list of elements described below, although these should only be accessed through accessor and replacement functions, not directly (since the internals may be subject to change). The structure of a corpus classed list object is:

`$documents`	A data frame containing the document level information, consisting of `texts`, user-named `docvars` variables describing attributes of the documents, and `metadoc` document-level metadata whose names begin with an underscore character, such as `_language`.
`$metadata`	A named list set of corpus-level meta-data, including `source` and `created` (both generated automatically unless assigned), `notes`, and `citation`.
`$settings`	Settings for the corpus which record options that govern the subsequent processing of the corpus when it is converted into a document-feature matrix (dfm). See settings.
`$tokens`	An indexed list of tokens and types tabulated by document, including information on positions. Not yet fully implemented.

See Also

docvars, metadoc, metacorpus, language, encoding, settings, texts

Examples

```
## Not run:
# import texts from a directory of files
corpus(directory("~/Dropbox/QUANTESS/corpora/ukManRenamed"),
       enc="UTF-8",
       source="Kens UK manifesto archive")

# choose a directory using a GUI
corpus(directory())
## End(Not run)
#
# create a corpus from texts
corpus(inaugTexts)

# create a corpus from texts and assign meta-data and document variables
uk2010immigCorpus <- corpus(uk2010immig,
                            docvars=data.frame(party=names(uk2010immig)),
                            enc="UTF-8")
```

`countSyllables` *Returns a count of the number of syllables in the input This function takes a text and returns a count of the number of syllables it contains. For British English words, the syllable count is exact and looked up from the CMU pronunciation dictionary. For any word not in the dictionary the syllable count is estimated by counting vowel clusters.*

Description

Returns a count of the number of syllables in the input This function takes a text and returns a count of the number of syllables it contains. For British English words, the syllable count is exact and looked up from the CMU pronunciation dictionary. For any word not in the dictionary the syllable count is estimated by counting vowel clusters.

Usage

```
countSyllables(sourceText)
```

Arguments

`sourceText`	Character vector of texts whose syllables will be counted

Details

This only works for English.

Value

numeric Named vector of counts of the number of syllables for each element of sourceText. When a word is not available in the lookup table, its syllables are estimated by counting the number of (English) vowels in the word.

Examples

```
countSyllables("This is an example sentence.")
myTexts <- c("Text one.", "Superduper text number two.", "One more for the road.")
names(myTexts) <- paste("myText", 1:3, sep="")
countSyllables(myTexts)
```

`describeTexts` *print a summary of texts Prints to the console a desription of the texts, including number of types, tokens, and sentences*

Description

print a summary of texts Prints to the console a desription of the texts, including number of types, tokens, and sentences

Usage

```
describeTexts(txts, verbose = TRUE)
```

Arguments

`txts`	The texts to be described
`verbose`	Default is TRUE. Set to false to suppress output messages

Examples

```
describeTexts(c("testing this text", "and this one"))
describeTexts(uk2010immig)
```

dfm	*Create a document-feature matrix from a corpus object*

Description

returns a document by feature matrix compatible with austin. A typical usage would be to produce a word-frequency matrix where the cells are counts of words by document.

Usage

```
dfm(x, ...)

## S3 method for class corpus
dfm(x, feature = c("word"), stem = FALSE,
  stopwords = NULL, bigram = FALSE, groups = NULL, verbose = TRUE,
  dictionary = NULL, dictionary_regex = FALSE, clean = TRUE,
  removeDigits = TRUE, removePunct = TRUE, lower = TRUE, addto = NULL,
  ...)

## S3 method for class character
dfm(x, feature = c("word"), stem = FALSE,
  stopwords = NULL, bigram = FALSE, verbose = TRUE, dictionary = NULL,
  dictionary_regex = FALSE, clean = TRUE, removeDigits = TRUE,
  removePunct = TRUE, lower = TRUE, addto = NULL, ...)

is.dfm(x)
```

Arguments

`x`	Corpus or character vector from which to generate the document-feature matrix
`feature`	Feature to count (e.g. words)
`stem`	Stem the words
`stopwords`	A character vector of stopwords that will be removed from the text when constructing the dfm. If `NULL` (default) then no stopwords will be applied. If "TRUE" then it currently defaults to `stopwords`.
`groups`	Grouping variable for aggregating documents
`verbose`	Get info to screen on the progress
`dictionary`	A list of character vector dictionary entries, including regular expressions (see examples)
`dictionary_regex`	`TRUE` means the dictionary is already in regular expression format, otherwise it will be converted from "wildcard" format
`addto`	`NULL` by default, but if an existing dfm object is specified, then the new dfm will be added to the one named. If both dfm's are built from dictionaries, the combined dfm will have its `Non_Dictionary` total adjusted.

Details

`is.dfm` returns `TRUE` if and only if its argument is a dfm.

Value

A matrix object with row names equal to the document names and column names equal to the feature labels. This matrix has `names(dimnames) = c("docs", "words")` to make it conformable to an wfm object.

Author(s)

Kenneth Benoit

Examples

```
data(inaugCorpus)
wfm <- dfm(inaugCorpus)

## by president, after 1960
wfmByPresfrom1900 <- dfm(subset(inaugCorpus, Year>1900), groups="President")
docnames(wfmByPresfrom1900)

## with dictionaries
data(iebudgets)
mycorpus <- subset(iebudgets, year==2010)
mydict <- list(christmas=c("Christmas", "Santa", "holiday"),
               opposition=c("Opposition", "reject", "notincorpus"),
               taxing="taxing",
               taxation="taxation",
               taxregex="tax*")
dictDfm <- dfm(mycorpus, dictionary=mydict)
dictDfm

## removing stopwords
testText <- "The quick brown fox named Seamus jumps over the lazy dog Rory, with Toms newpaper in his mout
testCorpus <- corpus(testText)
settings(testCorpus, "stopwords")
dfm(testCorpus, stopwords=TRUE)
if (require(tm)) {
}
```

`dfm2ldaformat` *Convert a quanteda `dfm` (document feature matrix) into a the data format needed by lda*

Description

Convert a quanteda `dfm` (document feature matrix) into a the data format needed by lda

Usage

```
dfm2ldaformat(d)
```

Arguments

d	A `dfm` object

Value

A list with components "documents" and "vocab" as needed by lda.collapsed.gibbs.sampler

Examples

```
data(inaugCorpus)
inaugCorpus <- subset(inaugCorpus, year>1960)
# create document-feature matrix, remove stopwords
d <- dfm(inaugCorpus, stopwords=TRUE)
# trim low frequency words
d <- dfmTrim(d, minCount=5, minDoc=3)
td <- dfm2ldaformat(d)
if (require(lda)) {
    tmodel.lda <- result <- lda.collapsed.gibbs.sampler(documents=td$documents,
                                                    K=10,
                                                    vocab=td$vocab,
                                                    num.iterations=50, alpha=0.1, eta=0.1)
}
top.topic.words(tmodel.lda$topics, 10, by.score=TRUE) # top five words in each topic
```

---

dfm2tmformat — *Convert a quanteda `dfm` (document feature matrix) into a* **tm** *DocumentTermMatrix*

Description

tm represents sparse document-feature matrixes in the simple triplet matrix format of the package **slam**. This function converts a `dfm` into a DocumentTermMatrix, for working with the `dfm` in **tm** or in other packages that expect this format, such as **topicmodels**.

Usage

```
dfm2tmformat(d, weighting = weightTf, ...)
```

Arguments

`d`	A dfm object
`weighting`	**tm**'s coercion function accepts weightings such as tf-idf, see **tm**'s as.DocumentTermMatrix for a list of possible arguments. The default is just tf (term frequency)

Value

A simple triplet matrix of class as.DocumentTermMatrix

Examples

```
data(inaugCorpus)
inaugCorpus <- subset(inaugCorpus, year==2010)
d <- dfmTrim(dfm(inaugCorpus), minCount=5, minDoc=3)
dim(d)
td <- dfm2tmformat(d)
length(td$v)
if (require(topicmodels)) tmodel.lda <- LDA(td, control = list(alpha = 0.1), k = 4)
```

`dfmSort` *sort a dfm by one or more margins*

Description

Sorts a dfm by documents or words

Usage

```
dfmSort(x, margin = c("words", "docs", "both"), decreasing = TRUE)
```

Arguments

`dfm`	Document-feature matrix created by `dfm`
`margin`	which margin to sort on `words` to sort words, `docs` to sort documents, and `both` to sort both
`decreasing`	TRUE (default) if sort will be in descending order

Value

A sorted dfm matrix object

Author(s)

Ken Benoit

Examples

```
data(inaugCorpus)
dtm <- dfm(inaugCorpus)
dtm[, 1:10]
dtm <- dfmSort(dtm, "words")
dfmSort(dtm)[, 1:10]
dfmSort(dtm, "both")[, 1:10]
```

`dfmTrim` *Trim a dfm based on a subset of features and words*

Description

Returns a document by feature matrix reduced in size based on document and term frequency, and/or subsampling.

Usage

```
dfmTrim(dfm, minCount = 5, minDoc = 5, sample = NULL, verbose = TRUE)
```

Arguments

`dfm`	Document-feature matrix created by `dfm`
`minCount`	minimum feature count
`minDoc`	minimum number of documents in which a feature appears
`sample`	how many features to retain (based on random selection)
`verbose`	print messages

Value

A dfm matrix object reduced in size.

Author(s)

Will Lowe, adapted by Ken Benoit

Examples

```
data(inaugCorpus)
dtm <- dfm(inaugCorpus)
dim(dtm)
dtmReduced <- dfmTrim(dtm, minCount=10, minDoc=2) # only words occuring at least 5 times and in at least 2d
dim(dtmReduced)
dtmSampled <- dfmTrim(dtm, sample=50)  # top 200 words
dim(dtmSampled)  # 196 x 200 words
```

`directory` *Function to declare a connection to a directory (containing files)*

Description

Function to declare a connection to a directory, although unlike file it does not require closing. If the directory does not exist, the function will return an error.

Usage

```
directory(path = NULL)
```

Arguments

`path`	String describing the full path of the directory or NULL to use a GUI to choose a directory from disk

Examples

```
## Not run:
# name a directory of files
mydir <- directory("~/Dropbox/QUANTESS/corpora/ukManRenamed")
corpus(mydir)

# choose a directory using a GUI
corpus(directory())
## End(Not run)
```

`docnames` *extract document names*

Description

Extract the document names from a corpus or a document-feature matrix. Document names are the rownames of the documents data.frame in a corpus, or the rownames of the dfm object for a dfm. of the dfm object.

docnames queries the document names of a corpus or a dfm

docnames <- assigns new values to the document names of a corpus. (Does not work for dfm objects, whose document names are fixed,)

Usage

```
docnames(x)

## S3 method for class corpus
docnames(x)

docnames(x) <- value

## S3 method for class dfm
docnames(x)
```

Value

docnames returns a character vector of the document names

docnames<- assigns a character vector of the document names in a corpus

Examples

```
# query the document names of the inaugural speech corpus
docnames(inaugCorpus) <- paste("Speech", 1:ndoc(inaugCorpus), sep="")

# reassign the document names of the inaugural speech corpus
docnames(inaugCorpus) <- paste("Speech", 1:ndoc(inaugCorpus), sep="")
#
# query the document names of a dfm
docnames(dfm(inaugTexts[1:5]))
```

`docvars` *get or set for document-level variables*

Description

Get or set variables for the documents in a corpus

Usage

```
docvars(x)

docvars(x, field) <- value
```

Arguments

`x`	corpus whose document-level variables will be read or set
`field`	string containing the document-level variable name

Value

`docvars` returns a data.frame of the document-level variables

`docvars<-` assigns `value` to the named `field`

Examples

```
head(docvars(inaugCorpus))
docvars(inaugCorpus, "President") <- paste("prez", 1:ndoc(inaugCorpus), sep="")
head(docvars(inaugCorpus))
```

`features.dfm` *extract the feature labels from a dfm*

Description

Extract the features from a document-feature matrix, which are stored as the column names of the dfm object.

Usage

```
## S3 method for class dfm
features(x)
```

Value

Character vector of the features

Examples

```
features(dfm(inaugTexts))[1:50]  # first 50 features (alphabetically sorted)
```

flatten.dictionary *Flatten a hierarchical dictionary into a list of character vectors*

Description

Converts a hierarchical dictionary (a named list of named lists, ending in character vectors at the lowest level) into a flat list of character vectors. Works like `unlist(dictionary, recursive=TRUE)` except that the recursion does not go to the bottom level.

Usage

```
flatten.dictionary(elms, parent = "", dict = list())
```

Arguments

`elms`	list to be flattened
`parent`	parent list name, gets built up through recursion in the same way that `unlist(dictionary, recurs` works
`dict`	the bottom list of dictionary entries ("synonyms") passed up from recursive calls

Details

Called by dfm()

Value

A dictionary flattened down one level further than the one passed

Author(s)

Kohei Watanabe

Examples

```
dictPopulismEN <-
    list(populism=c("elit*", "consensus*", "undemocratic*", "referend*",
                    "corrupt*", "propagand", "politici*", "*deceit*",
                    "*deceiv*", "*betray*", "shame*", "scandal*", "truth*",
                    "dishonest*", "establishm*", "ruling*"))
flatten.dictionary(dictPopulismEN)

hdict <- list(level1a = list(level1a1 = c("l1a11", "l1a12"),
                             level1a2 = c("l1a21", "l1a22")),
              level1b = list(level1b1 = c("l1b11", "l1b12"),
                             level1b2 = c("l1b21", "l1b22", "l1b23")),
              level1c = list(level1c1a = list(level1c1a1 = c("lowest1", "lowest2")),
                             level1c1b = list(level1c1b1 = c("lowestalone"))))
flatten.dictionary(hdict)
```

`getRootFileNames` *Truncate absolute filepaths to root filenames*

Description

This function takes an absolute filepath and returns just the document name

Usage

```
getRootFileNames(longFilenames)
```

Arguments

`longFilenames`	Absolute filenames including a full path with directory

Value

character vector of filenames withouth directory path

Author(s)

Paul Nulty

Examples

```
## Not run:
getRootFileNames(/home/paul/documents/libdem09.txt)

## End(Not run)
```

`getTextDir` *loads all text files from a given directory*

Description

given a directory name, get a list of all files in that directory and load them into a character vector using getTextFiles

Usage

```
getTextDir(dirname, enc = "detect", pattern = "\\.txt$")
```

Arguments

`dirname`	A directory path

Value

character vector of texts read from disk

Author(s)

Paul Nulty

Examples

```
## Not run:
getTextDir(/home/paul/documents/)

## End(Not run)
```

`getTextDirGui` *provides a gui interface to choose a gui to load texts from*

Description

launches a GUI to allow the user to choose a directory from which to load all files.

Usage

```
getTextDirGui()
```

Value

character vector of texts read from disk

Author(s)

Paul Nulty

Examples

```
## Not run:
getTextFiles(/home/paul/documents/libdem09.txt)

## End(Not run)
```

`getTextFiles` *load text files from disk into a vector of character vectors points to files, reads them into a character vector of the texts with optional names, default being filenames returns a named vector of complete, unedited texts*

Description

load text files from disk into a vector of character vectors points to files, reads them into a character vector of the texts with optional names, default being filenames returns a named vector of complete, unedited texts

Usage

```
getTextFiles(filenames, textnames = NULL, enc = "unknown",
  verbose = FALSE)
```

Arguments

`filenames`	a vector of paths to text files
`textnames`	names to assign to the texts
`verbose`	If TRUE, print out names of files being read. Default is FALSE

Value

character vector of texts read from disk

Author(s)

Paul Nulty

Examples

```
## Not run:
getTextFiles(/home/paul/documents/libdem09.txt)

## End(Not run)
```

`getWordStat` *Imports a Wordstat corpus from an XML file*

Description

Reads in a wordstat XML file and creates a corpus object with the document as text and variables as attributes

Usage

```
getWordStat(filename = NULL)
```

Arguments

`filename`	Path to wordstat XML file

getWordStatCSV — *Imports a Wordstat corpus from a CSV file*

Description

Reads in a wordstat CSV file and creates a corpus object with the document as text and variables as attributes

Usage

```
getWordStatCSV(filename = NULL)
```

Arguments

filename	Path to wordstat CSV file

inaugCorpus — *A corpus of US presidential inaugural addresses from 1789-2013*

Description

inaugCorpus is the quanteda corpus object of US presidents' inaugural addresses since 1789. Document variables contain the year of the address and the last name of the president.

inaugTexts is the character vector of US presidential inaugaration speeches

References

https://archive.org/details/Inaugural-Address-Corpus-1789-2009 and http://www.presidency.ucsb.edu/inaugurals.php.

Examples

```
# some operations on the inaugural corpus
data(inaugCorpus)
summary(inaugCorpus)
head(docvars(inaugCorpus), 10)
# working with the character vector only
data(inaugTexts)
str(inaugTexts)
head(docvars(inaugCorpus), 10)
mycorpus <- corpus(inaugTexts)
```

inaugTexts — *Texts of US presidential inaugaration speeches*

Description

Character vector of US presidential inaugaration speeches

kwic — *List key words in context from a text or a corpus of texts.*

Description

For a text or a collection of texts (in a quanteda corpus object), return a list of a keyword supplied by the user in its immediate context, identifying the source text and the word index number within the source text. (Not the line number, since the text may or may not be segmented using end-of-line delimiters.)

Usage

```
kwic(x, word, window = 5, regex = TRUE)

## S3 method for class character
kwic(x, word, window = 5, regex = TRUE)

## S3 method for class corpus
kwic(x, word, window = 5, regex = TRUE)
```

Arguments

`x`	A text character scalar or a quanteda corpus. (Currently does not support character vectors.)
`word`	A keyword chosen by the user.
`window`	The number of context words to be displayed around the keyword.
`regex`	If TRUE (default), then "word" is a regular expression, otherwise only match the whole word. Note that if regex=TRUE and no special regular expression characters are used in the search query, then the concordance will include all words in which the search term appears, and not just when it appears as an entire word. (For instance, searching for the word "key" will also return "whiskey".)
`texts`	a vector of texts
`corp`	a quanteda corpus object

Value

A data frame with the context before (`preword`), the keyword in its original format (`word`, preserving case and attached punctuation), and the context after (`postword`). The rows of the dataframe will be named with the word index position, or the text name and the index position for a corpus object.

Author(s)

Kenneth Benoit and Paul Nulty

Examples

```
kwic(inaugTexts, "terror")
kwic(inaugTexts, "terror", regex=FALSE)  # returns only whole word, without trailing punctuation
data(iebudgets)
kwic(subset(iebudgets, year==2010), "Christmas", window=4) # on a corpus
```

kwic2 *This function is an alternative KWIC*

Description

This function is an alternative KWIC

Usage

```
kwic2(texts, word, window = 30, filter = "", location = TRUE,
  case = TRUE)
```

Arguments

`text`	Texts
`word`	Word of interest
`window`	Window span in character
`filter`	Filter files in texts by regular expression
`location`	Show location of the word
`case`	Ignore case

Value

cfvm2 Collocatons as data frame

Author(s)

Kohei Watanabent

Examples

```
## Not run:
kwic2(texts, "we", filter = _2010, location=TRUE)

## End(Not run)
```

language *get or set the language of corpus documents*

Description

Get or set the `_language` document-level metadata field in a corpus. Same as

Usage

```
language(corp)
```

`likelihood.test` *likelihood test for 2x2 tables*

Description

returns a list of values

Usage

```
likelihood.test(x)
```

Arguments

`x`	a contingency table or matrix object

Value

A list of return values

Author(s)

Kenneth Benoit

`MCMCirtPoisson1d` *Bayesian-MCMC version of a 1-dimensional Poisson IRT scaling model*

Description

`MCMCirtPoisson1d` implements a flexible, Bayesian model estimated in JAGS using MCMC. It is based on the implementation of `wordfish` from the austin package. Options include specifying a model for alpha using document-level covariates, and partitioning the word parameters into different subsets, for instance, countries.

Usage

```
MCMCirtPoisson1d(dtm, dir = c(1, 2), control = list(sigma = 3, startparams =
  NULL), verbose = TRUE, itembase = 1, startRandom = FALSE, nChains = 1,
  nAdapt = 100, nUpdate = 300, nSamples = 200, nThin = 1, ...)
```

Arguments

`dtm`	The document-term matrix. Ideally, documents form the rows of this matrix and words the columns, although it should be correctly coerced into the correct shape.
`dir`	A two-element vector, enforcing direction constraints on theta and beta, which ensure that `theta[dir[1]] < theta[dir[2]]`. The elements of `dir` will index documents.

`control`	list specifies options for the estimation process. These are: `tol`, the proportional change in log likelihood sufficient to halt estimation, `sigma` the standard deviation for the beta prior in poisson form, and `startparams` a previously fitted wordfish model. `verbose` generates a running commentary during estimation. See `wordfish`.
`itembase`	A index or column name from `dtm` indicating which item should be used as the reference category. (These will have $\beta_j = 0$ and $\alpha_j = 0$.) The default is 1, to use the first category. If set to NULL then no constraints will be implemented. See details.
`verbose`	Turn this on for messages. Default is `TRUE`.
`startRandom`	`FALSE` by default, uses random starting values (good for multiple chains) if `TRUE`
`nChains`	Number of chains to run in JAGS.
`nAdapt`	Adaptation iterations in JAGS.
`nUpdate`	Update iterations in JAGS.
`nSamples`	Number of posterior samples to draw in JAGS.
`nThin`	Thinning parameter for drawing posterior samples in JAGS.
`...`	Additional arguments passed through.

Details

The ability to constrain an item is designed to make the additive Poisson GLM mathematically equivalent to the multinomial model for $R \times C$ contingency tables. We recommend setting a neutral category to have $\psi_0 = 0 and \beta_0 = 0$, for example the word "the" for a text count model (assuming this word has not been removed). Note: Currently the item-level return values will be returned in the original order suppled (`psi` and `beta`) but this is not true yet for the `mcmc.samples` value, which will have the constrained category as index 1. (We will fix this soon.)

Value

An augmented `wordfish` class object with additional stuff packed in. To be documented.

Author(s)

Kenneth Benoit

Examples

```
## Not run:
data(iebudgets)
# extract just the 2010 debates
iebudgets2010 <- subset(iebudgets, year==2010)

# create a document-term matrix and set the word margin to the columns
dtm <- dfm(iebudgets2010)

# estimate the maximium likelihood wordfish model from austin
require(austin)
iebudgets2010_wordfish <- wordfish(as.wfm(dtm, word.margin=2), dir=c(2,1))

# estimate the MCMC model, default values
iebudgets2010_wordfishMCMC <- MCMCirtPoisson1d(dtm, itembase="the", dir=c(2,1))
iebudgets2010_wordfishMCMC_unconstrained <- MCMCirtPoisson1d(dtm, dir=c(2,1))
```

```
# compare the estimates of \eqn{\theta_i}
require(psych)
pairs.panels(data.frame(ML=iebudgets2010_wordfish$theta,
                        PoissonThe=iebudgets2010_wordfishMCMC$theta,
                        PoissonUnconst=iebudgets2010_wordfishMCMC_unconstrained$theta),
             smooth=FALSE, scale=FALSE, ellipses=FALSE, lm=TRUE, cex.cor=2.5)
# inspect a known "opposition" word beta values
iebudgets2010_wordfish$beta[which(iebudgets2010_wordfishMCMC_unconstrained$words=="fianna")]
iebudgets2010_wordfishMCMC$beta[which(iebudgets2010_wordfishMCMC_unconstrained$words=="fianna")]
iebudgets2010_wordfishMCMC_unconstrained$beta[which(iebudgets2010_wordfishMCMC_unconstrained$words=="fianna

# random starting values, for three chains
dtm.sample <- trim(dtm, sample=200)
iebudgets2010_wordfishMCMC_sample <- MCMCirtPoisson1d(dtm.sample, dir=c(2,1), startRandom=TRUE, nChains=3)

## End(Not run)
```

---

metacorpus   *get or set corpus metadata*

Description

Get or set the corpus-level metadata in a quanteda corpus object.

Usage

```
metacorpus(corp, field = NULL)

metacorpus(corp, field) <- value
```

Arguments

`corp`	A quanteda corpus object
`field`	Metadata field name(s). If `NULL` (default), return all metadata names.

Value

For `metacorpus`, a list of the metadata fields in the corpus. If a list is not what you wanted, you can wrap the results in unlist, but this will remove any metadata field that is set to `NULL`.

For `metacorpus <-`, the corpus with the updated metadata.

Examples

```
metacorpus(inaugCorpus)
metacorpus(inaugCorpus, "source")
metacorpus(inaugCorpus, "citation") <- "Presidential Speeches Online Project (2014)."
metacorpus(inaugCorpus, "citation")
```

metadoc *get or set document-level meta-data*

Description

Get or set the document-level meta-data, including reserved fields for language and corpus.

Usage

```
metadoc(corp, field = NULL)
```

Arguments

`corp`	A quanteda corpus object

Value

For `texts`, a character vector of the texts in the corpus.

For `texts <-`, the corpus with the updated texts.

Note

Document-level meta-data names are preceded by an underscore character, such as `_encoding`, but when named in in the `field` argument, do *not* need the underscore character.

Examples

```
mycorp <- subset(inaugCorpus, Year>1990)
summary(mycorp, showmeta=TRUE)
metadoc(mycorp, "encoding") <- "UTF-8"
metadoc(mycorp)
metadoc(mycorp, "language") <- "english"
summary(mycorp, showmeta=TRUE)
```

naiveBayesText *Naive Bayes classifier for texts*

Description

Naive Bayes classifier for texts

Usage

```
naiveBayesText(x, y, smooth = 1, prior = "uniform",
  distribution = "multinomial", ...)
```

Arguments

`x`	character vector of training texts
`y`	character vector of test texts
`smooth`	smoothing parameter for feature counts by class
`prior`	prior distribution on texts, see details
`distribution`	count model for text features, can be `multinomial` or `Bernoulli`
`...`	

Details

Currently working for vectors of texts.

Value

A list of return values, consisting of:

`call`	original function call
`PwGc`	probability of the word given the class (empirical likelihood)
`Pc`	class prior probability
`PcGw`	posterior class probability given the word
`Pw`	baseline probability of the word
`data`	list consisting of `x` training class, and `y` test class
`distribution`	the distribution argument
`prior`	argument passed as a prior
`smooth`	smoothing parameter

Author(s)

Kenneth Benoit

`ndoc` *get the number of documents*

Description

Returns the number of documents in a corpus objects

Usage

```
## S3 method for class corpus
ndoc(corp)

## S3 method for class dfm
ndoc(x)
```

Arguments

`x`	a corpus or dfm object

Value

an integer (count) of the number of documents in the corpus or dfm

Examples

```
ndoc(inaugCorpus)
ndoc(dfm(inaugCorpus))
```

`ngrams` *Create ngrams*

Description

Create a set of ngrams (words in sequence) from a text.

Usage

```
ngrams(text, n = 2, concatenator = "_", include.all = FALSE, ...)
```

Arguments

`text`	character vector containing the texts from which ngrams will be extracted
`n`	the number of tokens to concatenate. Default is 2 for bigrams.
`window`	how many words to be counted for adjacency. Default is 1 for only immediately neighbouring words.
`concatenator`	character for combining words, default is _ (underscore) character
`include.all`	if TRUE, add n-1...1 grams to the returned list
`...`	additional arguments passed to tokenize

Value

a character vector of ngrams

Author(s)

Ken Benoit, Kohei Watanabe, Paul Nulty

Examples

```
ngrams("The quick brown fox jumped over the lazy dog.", n=2)
ngrams("The quick brown fox jumped over the lazy dog.", n=3)
ngrams("The quick brown fox jumped over the lazy dog.", n=3, concatenator="~")
ngrams("The quick brown fox jumped over the lazy dog.", n=3, include.all=TRUE)
```

`predict.naivebayes` *prediction method for Naive Bayes classifiers*

Description

prediction method for Naive Bayes classifier objects

Usage

```
## S3 method for class naivebayes
predict(object, newdata = NULL, scores = c(-1, 1))
```

Arguments

`object`	a naivebayes class object
`newdata`	new data on which to perform classification
`scores`	"reference" values when the wordscores equivalent implementation of Naive Bayes prediction is used. Default is `c(-1, 1)`.

Details

implements class predictions using trained Naive Bayes examples (from `naiveBayesText()`)

Value

A list of two data frames, named `docs` and `words` corresponding to word- and document-level predicted quantities

`docs`	data frame with document-level predictive quantities: nb.predicted, ws.predicted, bs.predicted, PcGw, wordscore.doc, bayesscore.doc, posterior.diff, posterior.logdiff. Note that the diff quantities are currently implemented only for two-class solutions.
`words`	data-frame with word-level predictive quantities: wordscore.word, bayesscore.word

Author(s)

Kenneth Benoit

`preprocess` *preprocess the tokens in a corpus*

Description

Applies pre-processing rules to the text and compiles a frequency table of features (word types) including counts of types, tokens, sentences, and paragraphs.

Usage

```
preprocess(corp)
```

Arguments

`corp`	Corpus to be preprocessed

Value

no return but modifies the object in place by changing

`tokens,`	a list consisting of the following:
`$dfm`	A dfm document-feature matrix object created with settings.
`$nwords`	A vector of token counts for each document.
`$ntypes`	A vector of type counts for each document.
`$nsents`	A vector of sentence counts for each document.
`$nparagr`	A vector of paragraph counts for each document.

Note

This will eventually become an indexing function. At the moment it creates and saves a dfm in addition to some summary information compiled from this, in order to speed up subsequent processing. Unlike most R functions which return a value, this one changes the object passed to it. (And they say R can't pass by reference...)

Examples

```
mycorpus <- corpus(uk2010immig)
mycorpus
preprocess(mycorpus)
mycorpus
mydfm <- dfm(mycorpus)
```

`quanteda` *An R package for the quantitative analysis of textual data.*

Description

A set of functions for creating and managing text corpora, extracting features from text corpora, and analyzing those features using quantitative methods.

Author(s)

Ken Benoit and Paul Nulty

`quantedaRefresh` *Re-install quanteda from github*

Description

Refresh the installation from the github repository for the package. Useful if you need to pull the latest changes.

Usage

```
quantedaRefresh(branch = c("dev", "master"))
```

Arguments

`branch`	default is `"dev"`

Value

Nothing

Author(s)

Kenneth Benoit

`readWStatDict` *Make a flattened list from a hierarchical wordstat dictionary*

Description

Make a flattened list from a hierarchical wordstat dictionary

Usage

```
readWStatDict(path)
```

Arguments

`path`	path to the wordstat dictionary file

Value

flattened dictionary as a list

`selectFeatures` *extract feature words This function takes type of feature extractor and a word freaquency matrix with binary class (1/0) to select features in class one. 'wsll' and 'wschisq' replicates of 'Keyness' of Wordsmith Tools.*

Description

extract feature words This function takes type of feature extractor and a word freaquency matrix with binary class (1/0) to select features in class one. 'wsll' and 'wschisq' replicates of 'Keyness' of Wordsmith Tools.

extract feature words This function takes type of feature extractor and a word freaquency matrix with binary class (1/0) to select features in class one. 'wsll' and 'wschisq' replicates of 'Keyness' of Wordsmith Tools.

Usage

```
selectFeatures(extractor, dfm, class, smooth = 1, show = 10)

selectFeatures(extractor, dfm, class, smooth = 1, show = 10)
```

Arguments

`extractor`	Type of feature extractor
`dfm`	Word frequency matrix
`class`	Biarny class
`smooth`	Smoothing constant
`show`	Number of features shown
`extractor`	Type of feature extractor
`dfm`	Word frequency matrix
`class`	Biarny class
`smooth`	Smoothing constant
`show`	Number of features shown

Value

data frame of feature words

data frame of feature words

Author(s)

Kohei Watanabe

Kohei Watanabe

Examples

```
## Not run:
texts <- getTextDir("/home/kohei/Documents/budget_2010/")
class  <- rep(0, length(texts))
class[grep("_LAB", names(texts))] <- 1
class[grep("_FF", names(texts))] <- 0
corpus <- corpusCreate(texts, attribs=list(class=class))
dfm <- dfm(corpus)
features <- selectFeatures(ll, dfm, corpus$attribs$class, smooth=1)

## End(Not run)
## Not run:
texts <- getTextDir("/home/kohei/Documents/budget_2010/")
class  <- rep(0, length(texts))
class[grep("_LAB", names(texts))] <- 1
class[grep("_FF", names(texts))] <- 0
corpus <- corpusCreate(texts, attribs=list(class=class))
dfm <- dfm(corpus)
features <- selectFeatures(ll, dfm, corpus$attribs$class, smooth=1)

## End(Not run)
```

`sentenceSeg` *split a text into sentences This function takes a text and splits it into sentences.*

Description

split a text into sentences This function takes a text and splits it into sentences.

Usage

```
sentenceSeg(text, pat = "[\\.\\?\\!][\\n* ]|\\n\\n*",
  abbreviations = NULL, stripempty = TRUE)
```

Arguments

`text`	Text to be segmented
`pat`	The regular expression for recognizing end of sentence delimiters.
`abbreviations`	A list of abbreviations'.' and therefore should not be used to segment text
`stripempty`	Remove empty "sentences", TRUE by default. Should only be set to false if for some reason you wanted to preserve the original text with all of its spaces etc.

Examples

```
test <- "This is a sentence! Several sentences. Its designed by a Dr. to test whether this function works.
sentenceSeg(test)
```

settings *Get or set the corpus settings*

Description

Get or set the corpus settings

Get or set various settings in the corpus for the treatment of texts, such as rules for stemming, stop-words, collocations, etc. `settings(corp)` query the corps settings `settings(corp, settingname) <-` update the corpus settings

Get the settings from a which a dfm was created

Usage

```
settings(x, ...)

## S3 method for class corpus
settings(corp, fields = NULL)

settings(corp, fields) <- value

## S3 method for class dfm
settings(x)
```

Arguments

`corp`	Corpus from/to which settings are queried or applied
`fields`	a valid corpus setting field name
`x`	dfm from which settings are queried

Examples

```
settings(tempcorpus, "stopwords")
tempdfm <- dfm(inaugCorpus)
tempdfmSW <- dfm(inaugCorpus, stopwords=TRUE)
settings(inaugCorpus, "stopwords") <- TRUE
tempdfmSW <- dfm(inaugCorpus)
tempdfm <- dfm(inaugCorpus, stem=TRUE)
settings(tempdfm)
```

sort.dfm *sort a dfm by one or more margins*

Description

Sorts a dfm by frequency of total features, total features in documents, or both

Usage

```
## S3 method for class dfm
sort(x, decreasing = TRUE, margin = c("features", "docs",
  "both"))
```

Arguments

`dfm`	Document-feature matrix created by `dfm`
`margin`	which margin to sort on `features` to sort by frequency of features, `docs` to sort by total feature counts in documents, and `both` to sort by both
`decreasing`	TRUE (default) if sort will be in descending order, otherwise sort in increasing order

Value

A sorted dfm matrix object

Author(s)

Ken Benoit

Examples

```
dtm <- dfm(inaugCorpus)
dtm[1:10, 1:5]
dtm <- sort(dtm)
sort(dtm)[1:10, 1:5]
sort(dtm, TRUE, "both")[1:10, 1:5]  # note that the decreasing=TRUE argument
                                    # must be second, because of the order of the
                                    # formals in the generic method of sort()
```

`stopwords` *A named list containing common stopwords in 14 languages*

Description

SMART English stopwords from the SMART information retrieval system (obtained from http://jmlr.csail.mit.edu/papers smart-stop-list/english.stop) and a set of stopword lists from the Snowball stemmer project in different languages (obtained from http://svn.tartarus.org/snowball/trunk/website/algorithms/*/stop.txt). Supported languages are danish, dutch, english, finnish, french, german, hungarian, italian, norwegian, portuguese, russian, spanish, and swedish. Language names are case sensitive. Alternatively, their IETF language tags may be used.

stopwordsGet *access stopwords*

Description

This function retrieves stopwords from the type specified in the `kind` argument and returns the stopword list as a character vector The default is English.

Usage

```
stopwordsGet(kind = "english")
```

Arguments

`kind`	The pre-set kind of stopwords (as a character string)

Value

a character vector or dfm with stopwords removed

Examples

```
stopwordsGet()
stopwordsGet("italian")
```

stopwordsRemove *remove stopwords from a text or dfm*

Description

This function takes a character vector or dfm and removes words in the remove common or 'semantically empty' words from a text.

Usage

```
stopwordsRemove(text, stopwords = NULL)

## S3 method for class character
stopwordsRemove(text, stopwords = NULL)

## S3 method for class matrix
stopwordsRemove(text, stopwords = NULL)
```

Arguments

`text`	Text from which stopwords will be removed
`stopwords`	Character vector of stopwords to remove

Details

This function takes a character vector 'text' and removes words in the list provided in 'stopwords'. If no list of stopwords is provided a default list for English is used.

Value

a character vector or dfm with stopwords removed

Examples

```
## examples for character objects
someText <- "Here is an example of text containing some stopwords we want to remove."
itText <- "Ecco un esempio di testo contenente alcune parole non significative che vogliamo rimuovere."
stopwordsRemove(someText)
stopwordsRemove(someText, stopwordsGet("SMART"))
stopwordsRemove(itText, stopwordsGet("italian"))
stopwordsRemove(someText, c("containing", "example"))

## example for dfm objects
data(iebudgets)
wfm <- dfm(subset(iebudgets, year==2010))
wfm.nostopwords <- stopwordsRemove(wfm)
dim(wfm)
dim(wfm.nostopwords)
dim(stopwordsRemove(wfm, stopwordsGet("SMART")))
```

`subset.corpus` *extract a subset of a corpus*

Description

Works just like the normal subset command but for corpus objects

Usage

```
## S3 method for class corpus
subset(corpus, subset = NULL, select = NULL)
```

Arguments

`corpus`	corpus object to be subsetted.
`subset`	logical expression indicating elements or rows to keep: missing values are taken as false.
`select`	expression, indicating the attributes to select from the corpus

Value

corpus object

Examples

```
## Not run:
data(inaugCorpus)
summary(subset(inaugCorpus, Year>1980))

## End(Not run)
```

`summary.corpus` *Corpus summary*

Description

Displays information about a corpus object, including attributes and metadata such as date of number of texts, creation and source.

Usage

```
## S3 method for class corpus
summary(corp, n = 100, verbose = TRUE, showmeta = FALSE)
```

Arguments

`corp`	corpus to be summarized
`n`	maximum number of texts to describe, default=100
`verbose`	FALSE to turn off printed output
`showmeta`	TRUE to include document-level meta-data

Examples

```
summary(inaugCorpus)
summary(inaugCorpus, n=10)
mycorpus <- corpus(uk2010immig, docvars=data.frame(party=names(uk2010immig)), enc="UTF-8")
summary(mycorpus, showmeta=TRUE)  # show the meta-data
mysummary <- summary(mycorpus, verbose=FALSE)  # (quietly) assign the results
mysummary$Types / mysummary$Tokens             # crude type-token ratio
```

`syllableCounts` *A named list mapping words to counts of their syllables*

Description

A named list mapping words to counts of their syllables, generated from the CMU pronunciation dictionary

References

http://www.speech.cs.cmu.edu/cgi-bin/cmudict

Examples

```
data(syllableCounts)
syllableCounts["sixths"]
syllableCounts["onomatopeia"]
```

`tagPos` *Returns a table of the occurrences of different parts of speech in a sentence This function takes a sentence and tags each word with it's part of speech using openNLP's POS tagger, then returns a table of the parts of speech*

Description

http://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

Usage

```
tagPos(sentence)
```

Arguments

`sentence`	Sentence to be tagged

Examples

```
## Not run:
tagPos("This is an example sentence with nouns and verbs for tagging.")

## End(Not run)
```

`texts` *get or set corpus texts*

Description

Get or replace the texts in a quanteda corpus object.

Usage

```
texts(corp)

texts(corp) <- value
```

Arguments

`corp`	A quanteda corpus object
`rownames`	If TRUE, overwrite the names of the documents with names from assigned object.

Value

For `texts`, a character vector of the texts in the corpus.

For `texts <-`, the corpus with the updated texts.

Examples

```
texts(inaugCorpus)[1]
sapply(texts(inaugCorpus), nchar)  # length in characters of the inaugual corpus texts

## this doesnt work yet - need to overload [ for this replacement function
# texts(inaugTexts)[55] <- "GW Bushs second inaugural address, the condensed version."
```

`tf` *normalizes the term frequencies a dfm*

Description

Returns a matrix of term weights, as a dfm object

Usage

```
tf(x)
```

Arguments

`dfm`	Document-feature matrix created by `dfm`

Value

A dfm matrix object where values are relative term proportions within the document

Author(s)

Ken Benoit

Examples

```
data(inaugCorpus)
dtm <- dfm(inaugCorpus)
dtm[1:10, 100:110]
tf(dtm)[1:10, 100:110]
```

`tfidf.dfm` *compute the tf-idf weights of a dfm*

Description

Returns a matrix of tf-idf weights, as a dfm object

Usage

```
tfidf.dfm(x, normalize = TRUE)
```

Arguments

`x`	document-feature matrix created by `dfm`
`normalize`	whether to normalize term frequency by document totals

Value

A dfm matrix object where values are tf-idf weights

Author(s)

Ken Benoit

Examples

```
data(inaugCorpus)
dtm <- dfm(inaugCorpus)
dtm[1:10, 100:110]
tfidf(dtm)[1:10, 100:110]
tfidf(dtm, normalize=FALSE)[1:10, 100:110]
```

`tfidf.dfm` *compute the tf-idf weights of a dfm*

Description

Returns a matrix of tf-idf weights, as a dfm object

Usage

```
## S3 method for class dfm
tfidf(x, normalize = TRUE)
```

Arguments

`x`	document-feature matrix created by `dfm`
`normalize`	whether to normalize term frequency by document totals

Value

A dfm matrix object where values are tf-idf weights

Author(s)

Ken Benoit

Examples

```
data(inaugCorpus)
dtm <- dfm(inaugCorpus)
dtm[1:10, 100:110]
tfidf(dtm)[1:10, 100:110]
tfidf(dtm, normalize=FALSE)[1:10, 100:110]
```

`tokenize` *tokenize a set of texts*

Description

Tokenize the texts from a character vector or from a corpus.

Usage

```
tokenize(x, ...)

## S3 method for class character
tokenize(text, clean = FALSE, simplify = FALSE)

## S3 method for class corpus
tokenize(corpus, ...)
```

Value

A list of length `ndoc(texts)` of the tokens found in each text.

`topfeatures` *list the most frequent features*

Description

List the most frequently occuring features.

Usage

```
topfeatures(x, n = 10, decreasing = TRUE)

## S3 method for class dfm
topfeatures(x, n = 10, decreasing = TRUE)
```

Value

A named numeric vector of feature counts, where the names are the feature labels.

Examples

```
topfeatures(dfm(inaugCorpus))
topfeatures(dfm(inaugCorpus, stopwords=TRUE))
# least frequent features
topfeatures(dfm(inaugCorpus), decreasing=FALSE)
```

`translate` *Send text to the google translate research API This function translates a text by sending it to the google translate API.*

Description

Send text to the google translate research API This function translates a text by sending it to the google translate API.

Usage

```
translate(sourceText, sourceLanguage, targetLanguage, key = NULL,
  verbose = FALSE)
```

Arguments

`sourceText`	Text to be translated
`sourceLanguage`	Language of the source text
`targetLanguage`	Language of the translated text
`key`	API key for Google Translate research API

Examples

```
## Not run: translation <- translate(original, fr, de, key=insertkeyhere)
```

`translate.corpus` *Send a corpus to the google translate research API This function translates a the texts in a corpus by sending them to the google translate API.*

Description

Send a corpus to the google translate research API This function translates a the texts in a corpus by sending them to the google translate API.

Usage

```
translate.corpus(corpus, targetlanguageString, textvar = "texts",
  languagevar = "language", key = NULL)
```

Arguments

`corpus`	corpus to be translated
`targetlanguageString`	Language of the source text
`languagevar`	Language of the translated text

Examples

```
## Not run:
translation <- translate(original, fr, de, key=insertkeyhere)

## End(Not run)
```

`trim.dfm` *Trim a dfm based on a subset of features and words*

Description

Returns a document by feature matrix reduced in size based on document and term frequency, and/or subsampling.

Usage

```
## S3 method for class dfm
trim(x, minCount = 5, minDoc = 5, sample = NULL,
  verbose = TRUE)
```

Arguments

`x`	document-feature matrix created by dfm
`minCount`	minimum feature count
`minDoc`	minimum number of documents in which a feature appears
`sample`	how many features to retain (based on random selection)
`verbose`	print messages

Value

A dfm object reduced in size.

Author(s)

Ken Benoit adapted from code by Will Lowe (see trim)

Examples

```
data(inaugCorpus)
dtm <- dfm(inaugCorpus)
dim(dtm)
dtmReduced <- trim(dtm, minCount=10, minDoc=2) # only words occuring at least 5 times and in at least 2docu
dim(dtmReduced)
dtmSampled <- trim(dtm, sample=200)  # top 200 words
dim(dtmSampled)  # 196 x 200 words
```

`twitterSearch` *work-in-progress from-scratch interface to Twitter search API*

Description

work-in-progress from-scratch interface to Twitter search API

Usage

```
twitterSearch()
```

`twitterStreamer` *work-in-progress interface to Twitter streaming API*

Description

work-in-progress interface to Twitter streaming API

Usage

```
twitterStreamer()
```

`twitterTerms` *make a corpus object from results of a twitter REST search*

Description

All of the attributes returned by the twitteR library call are included as attributes in the corpus. A oauth key is required, for further instruction about the oauth processs see: https://dev.twitter.com/apps/new and the twitteR documentation

Usage

```
twitterTerms(query, numResults = 50, key, cons_secret, token, access_secret)
```

Arguments

`query`	Search string for twitter
`numResults`	Number of results desired.
`key`	Number of results desired.
`key`	'your consumer key here'
`cons_secret`	'your consumer secret here'
`token`	'your access token here'
`access_secret`	'your access secret here'

Examples

```
## Not run:
twCorp <- twitterTerms(example, 10, key, cons_secret, token, access_secret)

## End(Not run)
```

---

uk2010immig *Immigration-related sections of 2010 UK party manifestos*

Description

Extracts from the election manifestos of 9 UK political parties from 2010, related to immigration or asylum-seekers.

Format

A named character vector of plain ASCII texts

Examples

```
data(uk2010immig)
uk2010immigCorpus <- corpus(uk2010immig, docvars=list(party=names(uk2010immig)))
language(uk2010immigCorpus) <- "english"
encoding(uk2010immigCorpus) <- "UTF-8"
summary(uk2010immigCorpus)
```

---

wordcloud.dfm *Plot a word cloud for a dfm*

Description

plots a document as a wordcloud of its features

Usage

```
## S3 method for class dfm
wordcloud(dfm, doc.index, ...)
```

Arguments

`dfm`	document-feature matrix created in quanteda
`document`	index of the document whose words will be plotted
`...`	additional arguments to pass to wordcloud

Value

None

Author(s)

Kenneth Benoit

Examples

```
data(iebudgets)
iebudgets2010 <- subset(iebudgets, year==2010)
wfm <- dfm(iebudgets2010, stopwords=TRUE)
wordcloudDfm(wfm, 1)  # plot the finance ministers speech as a wordcloud
```

`wordcloudDfm` *Plot a word cloud for a dfm*

Description

plots a document as a wordcloud of its features

Usage

```
wordcloudDfm(dfm, doc.index, ...)
```

Arguments

`dfm`	document-feature matrix created in quanteda
`document`	index of the document whose words will be plotted
`...`	additional arguments to pass to wordcloud

Value

None

Author(s)

Kenneth Benoit

Examples

```
data(iebudgets)
iebudgets2010 <- subset(iebudgets, year==2010)
wfm <- dfm(iebudgets2010, stopwords=TRUE)
wordcloudDfm(wfm, 1)  # plot the finance ministers speech as a wordcloud
```

`wordfishMCMC` *Bayesian-MCMC version of the "wordfish" Poisson scaling model*

Description

`wordfishMCMC` implements a flexible, Bayesian model estimated in JAGS using MCMC. It is based on the implementation of `wordfish` from the `austin` package. Options include specifying a model for alpha using document-level covariates, and partitioning the word parameters into different subsets, for instance, countries.

Usage

```
wordfishMCMC(dtm, dir = c(1, 2), control = list(sigma = 3, startparams =
  NULL), alphaModel = c("free", "logdoclength", "modelled"),
  alphaFormula = NULL, alphaData = NULL, wordPartition = NULL,
  betaPartition = FALSE, wordConstraints = NULL, verbose = TRUE,
  PoissonGLM = FALSE, nChains = 1, nAdapt = 100, nUpdate = 300,
  nSamples = 100, nThin = 1, ...)
```

Arguments

`dtm`	The document-term matrix. Ideally, documents form the rows of this matrix and words the columns, although it should be correctly coerced into the correct shape.
`dir`	A two-element vector, enforcing direction constraints on theta and beta, which ensure that theta[dir[1]] < theta[dir[2]]. The elements of `dir` will index documents.
`control`	list specifies options for the estimation process. These are: `tol`, the proportional change in log likelihood sufficient to halt estimatioe, `sigma` the standard deviation for the beta prior in poisson form, and `startparams` a previously fitted wordfish model. `verbose` generates a running commentary during estimation. See `austin::wordfish`.
`alphaModel`	`free` means the α_i is entirely estimated; `logdoclength` means the alpha is predicted with an expected value equal to the log of the document length in words, similar to an offset in a Poisson model with variable exposure; `modelled` allows you to specify a formula and covariates for α_i using `alphaFormula` and `alphaData`.
`alphaFormula`	Model formula for hierarchical model predicting α_i.
`alphaData`	Data to form the model matrix for the hierarchical model predicting α_i.
`wordPartition`	A vector equal in length to the documents that specifies a unique value partitioning the word parameters. For example, alpha could be a Boolean variable for `EU` to indicate that a document came from a country outside the EU or inside the EU. Or, it could be a factor variable indicating the name of the country (as long as there are multiple documents per country). Internally, `wordPartition` is coerced to a factor. `NULL` indicates that no paritioning of the word-level parameters will take place (default).
`betaPartition`	Boolean indicating that the β parameter should also be partitioned according to `wordPartition`.

`wordConstraints`	An index with a minimim length of 1, indicating which words will be set equal across the `wordPartition` factors. `NULL` if `is.null(wordPartition)` (default).
`verbose`	Turn this on for messages. Default is `TRUE`.
`nChains`	Number of chains to run in JAGS.
`nAdapt`	Adaptation iterations in JAGS.
`nUpdate`	Update iterations in JAGS.
`nSamples`	Number of posterior samples to draw in JAGS.
`nThin`	Thinning parameter for drawing posterior samples in JAGS.
`PoissonGLM`	Boolean denoting that the basic model should be estimated where log(alpha) is ~ dflat() as per The BUGS Book pp131-132
`...`	Additional arguments passed through.

Value

An augmented `wordfish` class object with additional stuff packed in. To be documented.

Author(s)

Kenneth Benoit

Examples

```
## Not run:
data(iebudgets)
# extract just the 2010 debates
iebudgets2010 <- corpus.subset(iebudgets, year==2010)

# create a document-term matrix and set the word margin to the columns
dtm <- create.fvm.corpus(iebudgets2010)
dtm <- wfm(t(dtm), word.margin=2)

# estimate the maximium likelihood wordfish model from austin
iebudgets2010_wordfish <- wordfish(dtm, dir=c(2,1))

# estimate the MCMC model, default values
iebudgets2010_wordfishMCMC <- wordfishMCMC(dtm, dir=c(2,1))

# compare the estimates of \eqn{\theta_i}
plot(iebudgets2010_wordfish$theta, iebudgets2010_wordfishMCMC$theta)

# MCMC with a partition of the word parameters according to govt and opposition
# (FF and Greens were in government in during the debate over the 2010 budget)
# set the constraint on word partitioned parameters to be the same for "the" and "and"
iebudgets2010_wordfishMCMC_govtopp <-
    wordfishMCMC(dtm, dir=c(2,1),
    wordPartition=(iebudgets2010$attribs$party=="FF" | iebudgets2010$attribs$party=="Green"),
    betaPartition=TRUE, wordConstraints=which(words(dtm)=="the"))

## End(Not run)
```

Index